The Art of Programming
Data Compression

Lecture 10

Homework Review

Write a program that finds all of the .jpg files in the current directory, reads each one, performs some sort of image processing with PIL on each, and writes each back to disk with "_proc" in its name, ie - image.jpg -> image_proc.jpg

Homework Review

```
from PIL import Image,ImageChops
import os

filenames=[fsp for fsp in os.listdir(".") if fsp[-4:] in
(".jpg",".JPG")]

for fsp in filenames:
    img=Image.open(fsp)
    img2=ImageChops.invert(img)
    img2.save(fsp[:-4]+"_proc.jpg")
```

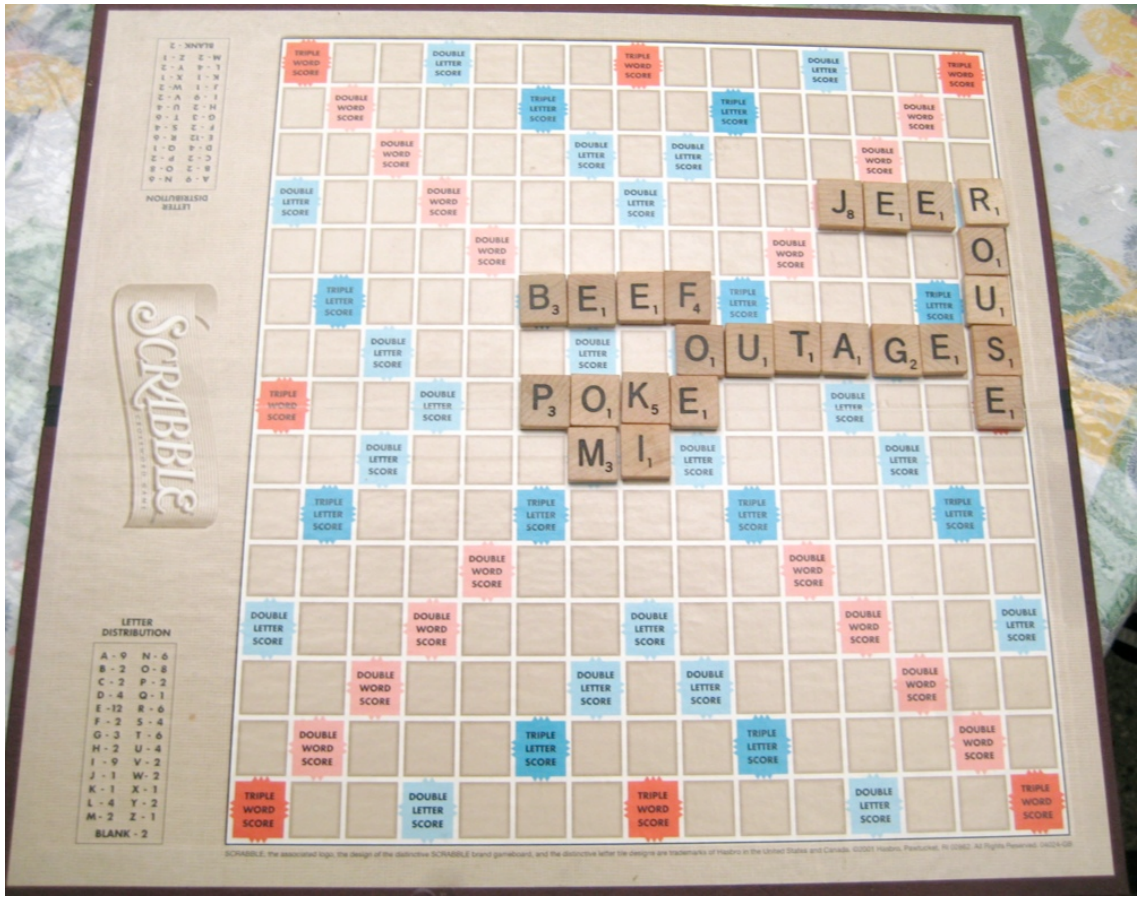
Jupyter Notebook Tip

```
from IPython.core.display import display, HTML
display(HTML("<style>.container { width:90% !important; }</style>"))
```

The Art of Programming

Different ways of doing the same thing

‘Scrabble’ Problem



What words could you make given these letters: PGAORRM ?

‘Scrabble’ Problem

- What words could you make given tiles containing PGAORRM
- 68 of them
- program armor gompa gramp groma maror morra pargo gamp gapo gora gorm gorp gram marg mora ogam orra parr pram prao proa prog prom ramp roam roar roma romp ago amp apo arm gam gap gar goa gor mag map mar moa mog mop mor oar ora pam par poa pom pro rag ram rap rom ag am ar go ma mo om op or pa po a

‘Scrabble’ Problem

- You have 7 random letters. What real words can you make from them ?
- How many ‘words’ could we make ?
- 7*6*5*4*3*2*1 + 7*6*5*4*3*2 + ... = 7! + 7!/1! + 7!/2! + 7!/3! + ... = 13699

‘Scrabble’ Problem

- You have 7 random letters. What real words can you make from them ?
- How many letter combinations could we make ?
- 7*6*5*4*3*2*1 + 7*6*5*4*3*2 + ... = 7! + 7!/1! + 7!/2! + 7!/3! + ... = 13699
- Different approaches:
  - Make all possible words, check to see if each is in the dictionary
    - Linear
    - Recursive
  - Check each word in the dictionary to see if it can be made from the letters in the list

Design #1

- ask for letters
- Read list of words
- Nested loop to make each possible word from letters
  - check to see if word is real
  - if so, add it to the list
- sort the list and print results

```
letters=input("Enter letters: ")

words=open("words.scrabble","r").readlines()
words=[i.strip().lower() for i in words]

# This generates all possible words and checks each to see if it's in the words list
goodwords=[]
n=[0]*7
for n[0] in range(7):
     w=letters[n[0]]
     if w in words : goodwords.append((len(w),w))  # 1 letter words
     for n[1] in range(7):
          if len(set(n[:2]))!=2 : continue         # we skip possibilities with duplicates
          w="".join([letters[j] for j in n[:2]])        # 2 letter words
          if w in words : goodwords.append((len(w),w))
          for n[2] in range(7):
               if len(set(n[:3]))!=3 : continue         # we skip possibilities with duplicates
               w="".join([letters[j] for j in n[:3]])        # 3 letter words
               if w in words : goodwords.append((len(w),w))
               for n[3] in range(7):
                    if len(set(n[:4]))!=4 : continue         # we skip possibilities with duplicates
                    w="".join([letters[j] for j in n[:4]])        # 4 letter words
                    if w in words : goodwords.append((len(w),w))
                    for n[4] in range(7):
                         if len(set(n[:5]))!=5 : continue         # we skip possibilities with duplicates
                         w="".join([letters[j] for j in n[:5]])        # 5 letter words
                         if w in words : goodwords.append((len(w),w))
                         for n[5] in range(7):
                              if len(set(n[:6]))!=6 : continue         # we skip possibilities with duplicates
                              w="".join([letters[j] for j in n[:6]])        # 6 letter words
                              if w in words : goodwords.append((len(w),w))
                              for n[6] in range(7):
                                   if len(set(n))!=7 : continue        # we skip possibilities with duplicates
                                   w="".join([letters[j] for j in n])       # 7 letter words
                                   if w in words : goodwords.append((len(w),w))


for x in reversed(sorted(set(goodwords))): print(x[1])
```

Profiling

- Spyder has a profiler
- 3 built-in profilers
  - profile, cProfile, hotshot
  - python -mcProfile script.py
- line_profiler
- ‘C’ level profilers
  - valgrind/cachegrind (linux)
  - dtrace (mac)

Profiling in Jupyter

- `%prun function()`

or

- conda install line-profiler
- in anaconda:
  - `%load_ext line_profiler`
  - `%timeit function()`
  - `%lprun -f function()`

```
letters=input("Enter letters: ")

words=open("words.scrabble","r").readlines()
words=[i.strip().lower() for i in words]
words=set(words)

# This generates all possible words and checks each to see if it's in the words list
goodwords=[]
n=[0]*7
for n[0] in range(7):
     w=letters[n[0]]
     if w in words : goodwords.append((len(w),w))  # 1 letter words
     for n[1] in range(7):
          if len(set(n[:2]))!=2 : continue         # we skip possibilities with duplicates
          w="".join([letters[j] for j in n[:2]])        # 2 letter words
          if w in words : goodwords.append((len(w),w))
          for n[2] in range(7):
               if len(set(n[:3]))!=3 : continue         # we skip possibilities with duplicates
               w="".join([letters[j] for j in n[:3]])        # 3 letter words
               if w in words : goodwords.append((len(w),w))
               for n[3] in range(7):
                    if len(set(n[:4]))!=4 : continue         # we skip possibilities with duplicates
                    w="".join([letters[j] for j in n[:4]])        # 4 letter words
                    if w in words : goodwords.append((len(w),w))
                    for n[4] in range(7):
                         if len(set(n[:5]))!=5 : continue         # we skip possibilities with duplicates
                         w="".join([letters[j] for j in n[:5]])        # 5 letter words
                         if w in words : goodwords.append((len(w),w))
                         for n[5] in range(7):
                              if len(set(n[:6]))!=6 : continue         # we skip possibilities with duplicates
                              w="".join([letters[j] for j in n[:6]])        # 6 letter words
                              if w in words : goodwords.append((len(w),w))
                              for n[6] in range(7):
                                   if len(set(n))!=7 : continue        # we skip possibilities with duplicates
                                   w="".join([letters[j] for j in n])       # 7 letter words
                                   if w in words : goodwords.append((len(w),w))


for x in reversed(sorted(set(goodwords))): print(x[1])
```

Design #1

We still have a potential problem:

- Fixed number of letters, what if we have 5 or 8 ?

Recursion

- Normal function:

```
def factorial(x):
    r=1
    for i in range(2,x):
        r*=i

    return r
```

- A function that calls itself

```
def factorial(x):
    if x==1 : return 1
    return x*factorial(x-1)
```

Design #2

- ask for letters
- Read list of words
- Recursion to make each possible word from letters
  - check to see if word is real
  - if so, add it to the list
- sort the list and print results

```
def all_good_words(letters,words,result,partial=""):
    """Generates all possible words using letters from set 'words' adds them to a list.
    takes a list of letters, possible words, and an empty list which will be
    modified in-place with results
    as (len,word) tuples."""

    if len(letters)==0 : return       # end of the recursion

    for l in letters:
        possible=partial+l
        if possible in words : result.append((len(possible),possible))

        shorter=list(letters) # note that this copies the list if it's already a list
        shorter.remove(l)
        all_good_words(shorter,words,result,possible)

letters=input("Enter letters: ")

words=open("words.scrabble","r").readlines()
words=set([i.strip().lower() for i in words])

# This generates all possible words and checks each to see if it's in the words list
result=[]
all_good_words(letters,words,result)

for x in reversed(sorted(set(result))): print(x[1])
```

Design #3

- ask for letters
- Read list of words
- Loop over list of words
  - make a list of available letters
  - Loop over the letters in word
    - see if each letter is in the list, if so, remove
  - if we found all of the letters print the word

```
def goodword(word,letters):
 tmp=list(letters)
 for letter in word:
  if letter in tmp:
    tmp.remove(letter)
    continue
  if '?' in tmp :
    tmp.remove('?')
    continue
  return False
 return True

letters=input("Enter letters: ")

words=open("twl.txt","r").readlines()
words=[i.strip() for i in words]

goodwords=[(len(x),x) for x in words if goodword(x,letters)]

for x in reversed(sorted(goodwords)):
 print(x[1])
```

Results

- Nested loop : 42 sec
- Nested loop (set): 0.44 sec
- Recursive : 0.38 sec
- Check all words: 0.92 sec

Prime Numbers

Find all prime numbers <100000

- How do we represent the data ?
  - Build a list containing primes
- Break the task into small pieces
  - Loop over all possible primes
    - Check if prime
      - Check all possible factors
    - If so, add to list
- Code each of the pieces

- Check to see if each number is divisible by every other number

```
for i in range(100000):
    for j in range(2,i):
        if i%j==0 : break
    else: print(i)
```

```
from time import time
t0=time()
primes=[]

for i in range(100000):
    for j in range(2,i):
        if i%j==0 : break
    else: primes.append(i)

t1=time()
print(primes[:10],primes[-10:])
print(t1-t0)
```

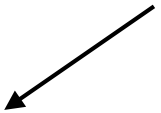
```
92.5 sec
```

```
from time import time
t0=time()
primes=[]

for i in range(100000):
    for j in range(2,i):
        if i%j==0 : break
    else: primes.append(i)

t1=time()
print(primes[:10],primes[-10:])
print(t1-t0)
```

Scales as $O(n^2)$



```
92.5 sec
```

- What if we skip even numbers, which are divisible by 2, as well as even factors, which cannot be prime

```
from time import time
t0=time()
primes=[1,2,3]

for i in range(5,100000,2):
    for j in range(3,i//2,2):
        if i%j==0 : break
    else: primes.append(i)

t1=time()
print(primes[:10],primes[-10:])
print(t1-t0)
```

```
22.2 sec
```

- Actually, we just need to find the FIRST prime factor to show that it isn't prime. The first prime factor, if it exists, is guaranteed to be <sqrt(number). Think about it...

```
from time import time
t0=time()
primes=[1,2,3]

for i in range(5,100000,2):
    for j in range(3,int(i**.5)+1,2):
        if i%j==0 : break
    else: primes.append(i)

t1=time()
print(primes[:10],primes[-10:])
print(t1-t0)
```

Scales as $O(n^{1.5})$

```
0.28 sec
```

- The only factors we need to check are themselves prime numbers.

```
from time import time
t0=time()
primes=[1,2,3]

for i in range(5,100000,2):
    for j in primes[2:]:
        if i%j==0 : break
    else: primes.append(i)

t1=time()
print(primes[:10],primes[-10:])
print(t1-t0)
```

```
9.3 sec
```

- Oops, forgot about the sqrt(i) limit

```
from time import time
t0=time()
primes=[1,2,3]

m=1
for i in range(5,100000,2):
    while primes[m-1]**2<i : m+=1
    for j in primes[2:m]:
        if i%j==0 : break
    else: primes.append(i)

t1=time()
print(primes[:10],primes[-10:])
print(t1-t0)
```

```
0.18 sec
```

Is there a completely different approach?

Sieve of Eratosthenes (~200 BC)

- remove non-primes:

```
from time import time
t0=time()

primes=set(range(100001))
for i in range(2,317):
    bad=range(i*2,100001,i)
    primes.difference_update(bad)

t1=time()
primes=list(sorted(primes))
print(primes[:10],primes[-10:])
print(t1-t0)
0.03 sec
```

- A 1-line program which does the same thing. Not the fastest, and certainly not easy to follow.

```
[i for i in range(3,100000,2) if len([j for j in
                    range(3,int(i**.5)+1,2) if i%j==0])==0]
```

Obfuscated C Contest

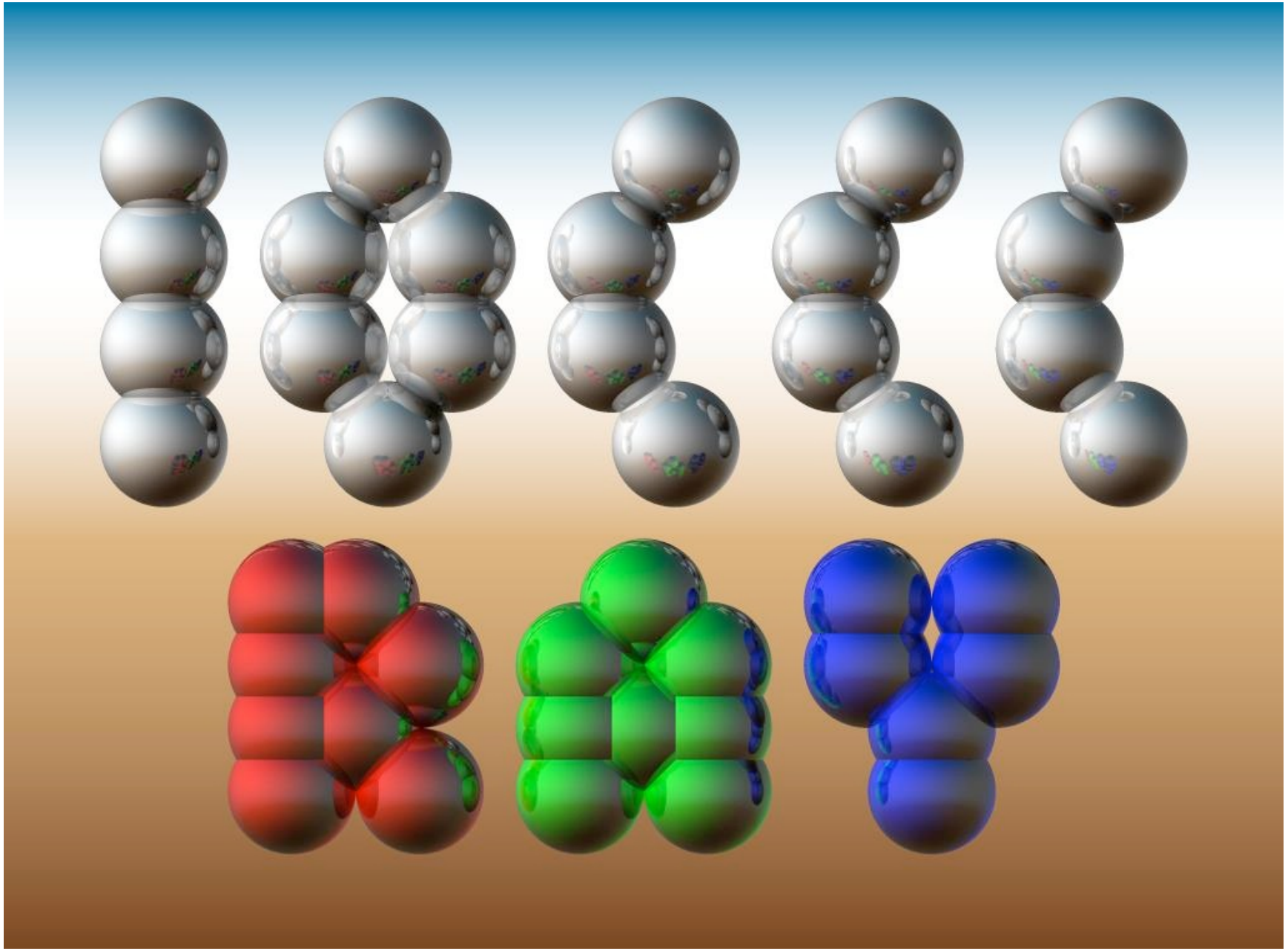
Goal of the competition is to produce a C program which does something interesting but is as difficult to read as possible, and may be internally complicated. This is much like the poetry of programming. Here is an example of a recent winner:

Obfuscated C Contest

```
                                X=1024; Y=768; A=3;

J=0;K=-10;L=-7;M=1296;N=36;O=255;P=9;_=1<<15;E;S;C;D;F(b){E="1""111886:6:??AAF"
"FHHMMOO55557799@@>>>BBBGGIIKK"[b]-64;C="C@=::C@@==@=:C@=:C@=:C5""31/513/5131/"
"31/531/53"[b ]-64;S=b<22?9:0;D=2;}I(x,Y,X){Y?(X^=Y,X*X>x?(X^=Y):0,  I (x,Y/2,X
)):(E=X);       }H(x){I(x,     _,0);}p;q(         c,x,y,z,k,l,m,a,           b){F(c
);x-=E*M     ;y-=S*M           ;z-=C*M          ;b=x*        x/M+          y*y/M+z
*z/M-D*D    *M;a=-x              *k/M     -y*l/M-z         *m/M;    p=((b=a*a/M-
b)>=0?(I    (b*M,_       ,0),b    =E,       a+(a>b      ?-b:b)):     -1.0);}Z;W;o
(c,x,y,     z,k,l,    m,a){Z=!    c?       -1:Z;c     <44?(q(c,x          ,y,z,k,
l,m,0,0     ),(p>      0&&c!=     a&&         (p<W          ||Z<0)           )?(W=
p,Z=c):     0,o(c+          1,    x,y,z,          k,l,           m,a)):0     ;}Q;T;
U;u;v;w    ;n(e,f,g,             h,i,j,d,a,    b,V){o(0       ,e,f,g,h,i,j,a);d>0
&&Z>=0? (e+=h*W/M,f+=i*W/M,g+=j*W/M,F(Z),u=e-E*M,v=f-S*M,w=g-C*M,b=(-2*u-2*v+w)
/3,H(u*u+v*v+w*w),b/=D,b*=b,b*=200,b/=(M*M),V=Z,E!=0?(u=-u*M/E,v=-v*M/E,w=-w*M/
E):0,E=(h*u+i*v+j*w)/M,h-=u*E/(M/2),i-=v*E/(M/2),j-=w*E/(M/2),n(e,f,g,h,i,j,d-1
,Z,0,0),Q/=2,T/=2,       U/=2,V=V<22?7:  (V<30?1:(V<38?2:(V<44?4:(V==44?6:3))))
,Q+=V&1?b:0,T                +=V&2?b         :0,U+=V    &4?b:0)      :(d==P?(g+=2
,j=g>0?g/8:g/     20):0,j    >0?(U=     j    *j/M,Q      =255-    250*U/M,T=255
-150*U/M,U=255    -100    *U/M):(U    =j*j     /M,U<M           /5?(Q=255-210*U
/M,T=255-435*U            /M,U=255    -720*      U/M):(U       -=M/5,Q=213-110*U
/M,T=168-113*U    /       M,U=111                -85*U/M)       ),d!=P?(Q/=2,T/=2
,U/=2):0);Q=Q<    0?0:      Q>O?     O:          Q;T=T<0?    0:T>O?O:T;U=U<0?0:
U>O?O:U;}R;G;B    ;t(x,y      ,a,    b){n(M*J+M    *40*(A*x   +a)/X/A-M*20,M*K,M
*L-M*30*(A*y+b)/Y/A+M*15,0,M,0,P,  -1,0,0);R+=Q    ;G+=T;B   +=U;++a<A?t(x,y,a,
b):(++b<A?t(x,y,0,b):0);}r(x,y){R=G=B=0;t(x,y,0,0);x<X?(printf("%c%c%c",R/A/A,G
/A/A,B/A/A),r(x+1,y)):0;}s(y){r(0,--y?s(y),y:y);}main(){printf("P6\n%i %i\n255"
                              "\n",X,Y);s(Y);}
```

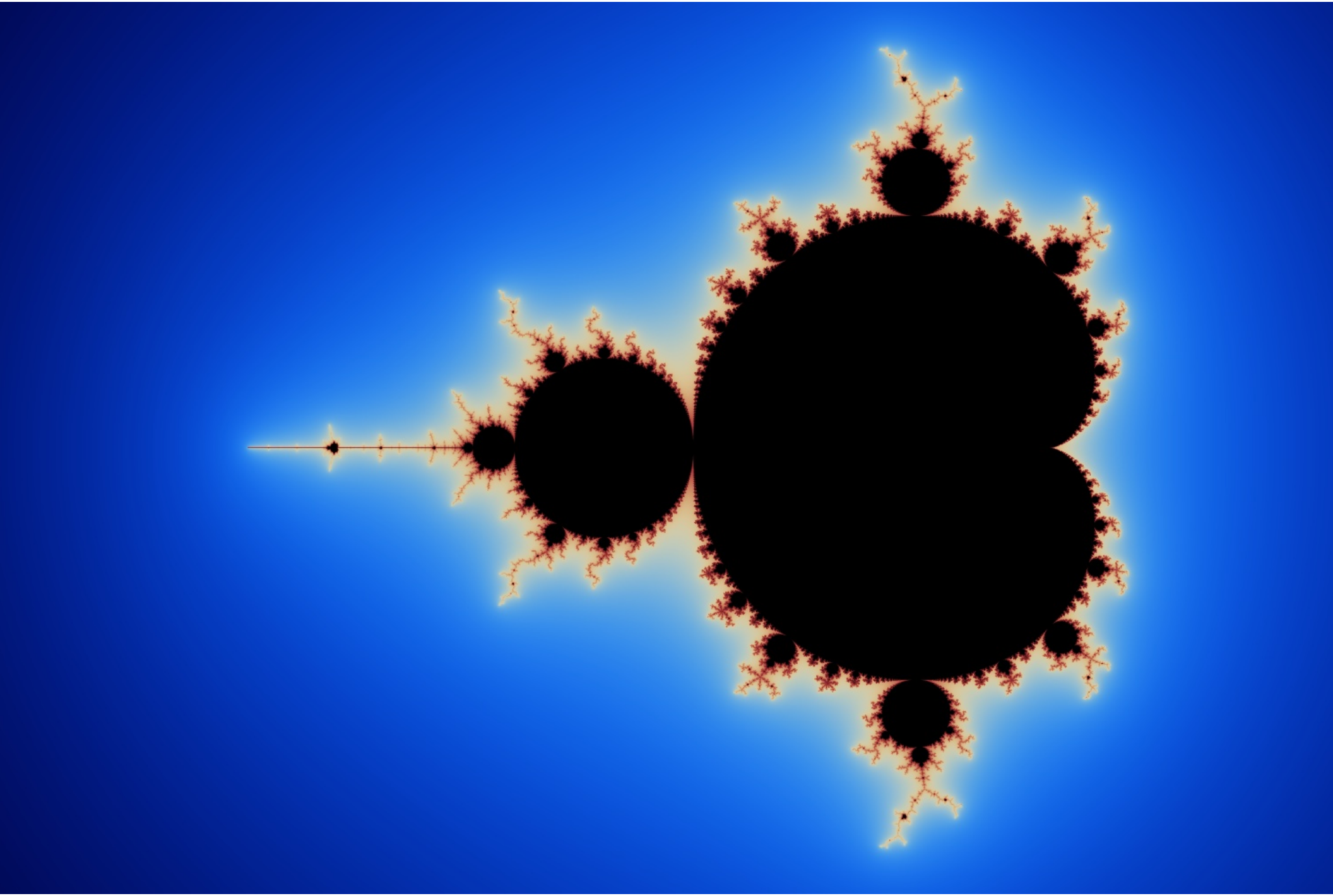
Obfuscated C Contest



Obfuscated Python ?

```
_                                       =   (
                                         255,
                                       lambda
                                  V       ,B,c
                                :c   and Y(V*V+B,B,  c
                                  -1)if(abs(V)<6)else
                        (          2+c-4*abs(V)**-0.4)/i
                          )  ;v,     x=1500,1000;C=range(v*x
                           );import  struct;P=struct.pack;M,\
                    j  ='<QIIHHHH',open('M.bmp','wb').write
for X in j('BM'+P(M,v*x*3+26,26,12,v,x,1,24))or C:
                    i  ,Y=_;j(P('BBB',*(lambda T:(T*80+T**9
                           *i-950*T  **99,T*70-880*T**18+701*
                          T  **9     ,T*i**(1-T**45*2)))(sum(
                        [             Y(0,(A%3/3.+X%v+(X/v+
                                       A/3/3.-x/2)/1j)*2.5
                                     /x   -2.7,i)**2 for  \
                                       A       in C
                                             [:9]])
                                               /9)
                                              )   )
```

note: this is Python 2 code



Debugging

- print statements ?
- traceback module: print_exc()

Debuggers

- Jupyter:
  - import pdb; pdb.set_trace()
- Spyder in Anaconda, built in editor/debugger!

- Built in IDLE
- https://wiki.python.org/moin/PythonDebuggingTools

DNA Sequence

- seq=b"atggcagctaaagacgtaaaattcggtaacgacgctcgtgtgaaaatgctgcgcggcgtaaacgtactggcagatgc
  agtgaaagttaccctcggtccgaaaggccgtaacgtagttctggataaatctttcggtgcaccgaccatcaccaaagatggtgtttc
  cgttgctcgtgaaatcgaactggaagacaagttcgaaaacatgggtgcgcagatggtgaaagaagttgcctctaaagcgaacg
  acgctgcaggcgacggtaccaccactgcaaccgtactggctcaggctatcatcactgaaggtctgaaagctgttgctgcgggc
  atgaacccgatggacctgaaacgtggtatcgacaaagctgttaccgctgcagttgaagaactgaaagcgctgtccgtaccgtgct
  ctgactctaaagcgattgctcaggttggtactatctccgctaactccgacgaaaccgtaggtaaactgatcgctgaagcgatggac
  aaagtcggtaaagaaggcgttatcaccgttgaagacggtaccggtctgcaggacgaactggacgtggttgaaggtatgcagttc
  gaccgtggctacctgtctccttacttcatcaacaagccggaaactggcgcagtagaactggaaagcccgttcatcctgctggctg
  acaagaaaatctccaacatccgcgaaatgctgccggttctggaagccgttgccaaagcaggcaaaccgctgctgatcatcgct
  gaagatgtagaaggcgaagcgctggcaactctggttgttaacaccatgcgtggcatcgtgaaagttgctgcagttaaagctccgg
  gcttcggcgatcgtcgtaaagctatgctgcaggatatcgcaaccctgactggcggtaccgtaatctctgaagagatcggtatgga
  gctggaaaaagcaaccctggaagacctgggtcaggctaaacgcgttgtgatcaacaaagacaccaccaccatcatcgatggc
  gtgggcgaagaagctgcaatccagggccgtgttgctcagatccgtcagcagattgaagaagcaacttctgactacgaccgtga
  aaaactgcaggagcgcgtagcgaaactggcaggcggcgttgcagttatcaaagtaggtgctgctaccgaagttgaaatgaaag
  agaaaaaagcacgcgttgaagacgccctgcacgcgacccgtgctgcggtagaagaaggcgtggttgctggtggtggtgttgc
  gctgatccgcgtagcgtctaaactggctgacctgcgtggtcagaacgaagaccagaacgtgggtatcaaagttgcactgcgtgc
  aatggaagctccgctgcgtcagatcgtcctgaactgcggcgaagaaccgtctgttgttgctaacaccgttaaaggcggcgacgg
  caactacggttacaacgcagcaaccgaagaatacggcaacatgatcgacatgggtatcctggacccaaccaaagtaacccgtt
  ctgctctgcagtacgcggcttctgtggctggcctgatgatcaccaccgaatgcatggttaccgacctgccgaaaaacgatgcagc
  tgacttaggcgctgctggcggtatgggcggcatgggtggcatgggcggcatgatgtaa"

Data Compression

- command line: gzip, zip, bzip2, …
- from zlib import compress,decompress
- compress(<bytes>,<level>)
  - compress(bytes(<str>,"utf-8"),<level>)
- decompress(<cmpbytes>)
- great, but…

Compress DNA Seq

- With gzip, etc.
  - Slooow, 2.8 GB gator genome takes 3 min (28%)
  - No random access of compressed data
- Can we come up with something better ?

Compress DNA Seq

- DNA, only 4 possible values -> 2 bits
- 1 Byte = 8 bits -> 4x compression !!!
- But…
- What about unknowns ?
- 5*5*5 = 125
  - 3x compression
  - naturally ordered into triplets

```
import os

letmap={"a":1,"c":2,"g":3,"t":4,"n":0}

# build compression/decompression dict
tripmap={}
tripmapinv={}
for a in letmap.keys():
    for b in letmap.keys():
        for c in letmap.keys():
            val=chr(letmap[a]*25+letmap[b]*5+letmap[c])
            tripmap[a+b+c]=val
            tripmapinv[val]=a+b+c

seq=open("/tmp/gator_genome.seq","r")
outseq=open("/tmp/gator_genome.cmp","w")

while True:
    trip=seq.read(3).lower()
    if len(trip)==0 : break
    try: outseq.write(tripmap[trip])
    except: break
```

```
#include <stdio.h>

int main(int argc, char *argv[]) {
    FILE *in,*out;

    in=fopen("/Users/stevel/Downloads/gator_small.seq","r");
    out=fopen("/Users/stevel/Downloads/gator_small.cmp","w");

    char seq[3];
    char map[256];
    map['a']=1;
    map['c']=2;
    map['g']=3;
    map['t']=4;
    map['n']=0;
    while (fread(seq,3,1,in)==1) {
        unsigned char xlate=map[seq[0]]*25+map[seq[1]]*5+map[seq[2]];
        fwrite(&xlate,1,1,out);
    }
    fclose(in);
    fclose(out);
}
```

Homework

- Make sure you have an SSH client available on your computer.
  - Mac/Linux, comes with the OS, nothing to do
  - Windows, a number of choices, this one is good:
  - http://www.chiark.greenend.org.uk/~sgtatham/putty/